# LAPORAN PRAKTIKUM SISTEM OPERASI MODUL 11: PENJADWALAN PROSES



Disusun oleh: Nama : Jesika Anjarwati

NIM : L200210258

Kelas : Praktikum SO E

# PROGRAM STUDI TEKNIK INFORMATIKA FAKULTAS KOMUNIKASI DAN INFORMATIKA

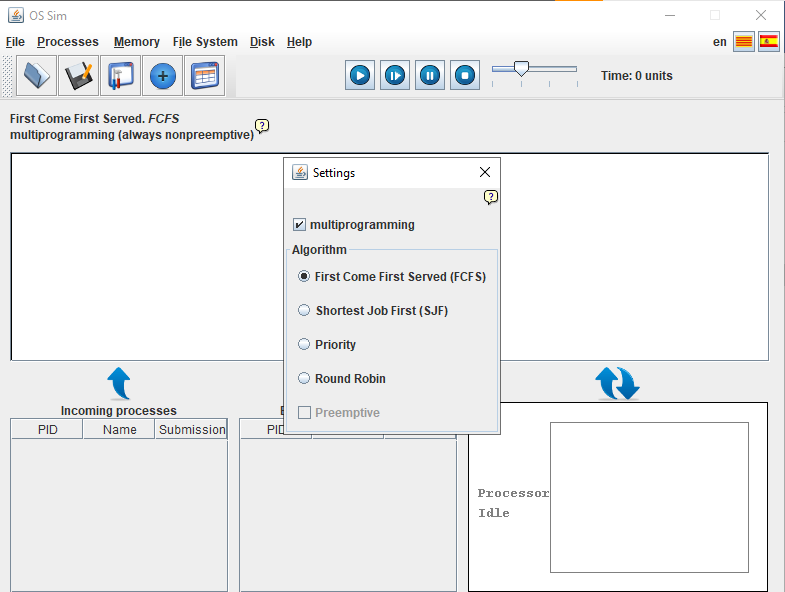
**UNIVERSITAS MUHAMMADIYAH SURAKARTA TAHUN 2022/2023**

## LANGKAH KERJA

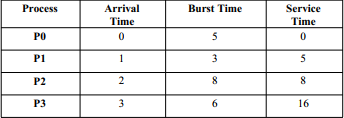
**Kegiatan 1. Penjadawalan Proses**

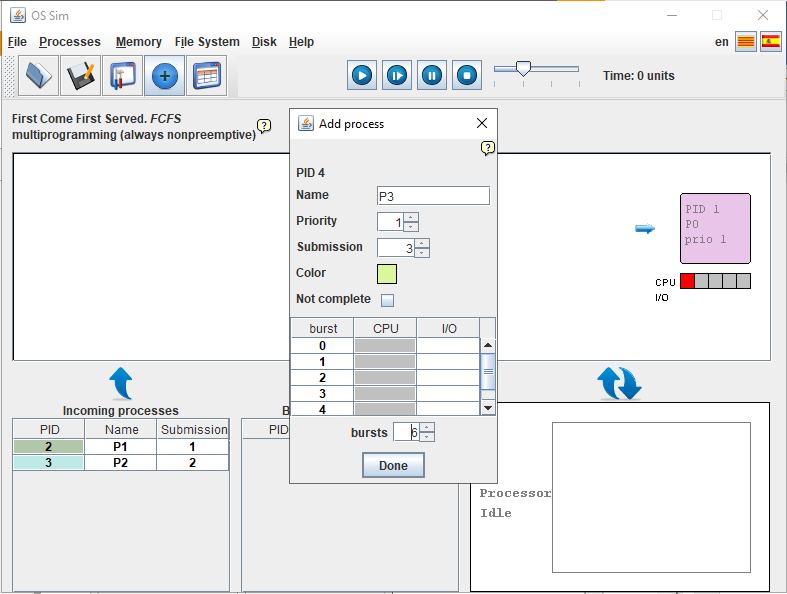
## First-Come, First-Served (FCFS)

* + 1. Bukalah program OSSim, selanjutnya pilih menu processes  process scheduling.
    2. Selanjutnya pilihlah setting dan pilih algoritma **First-Come, First-Served (FCFS).**

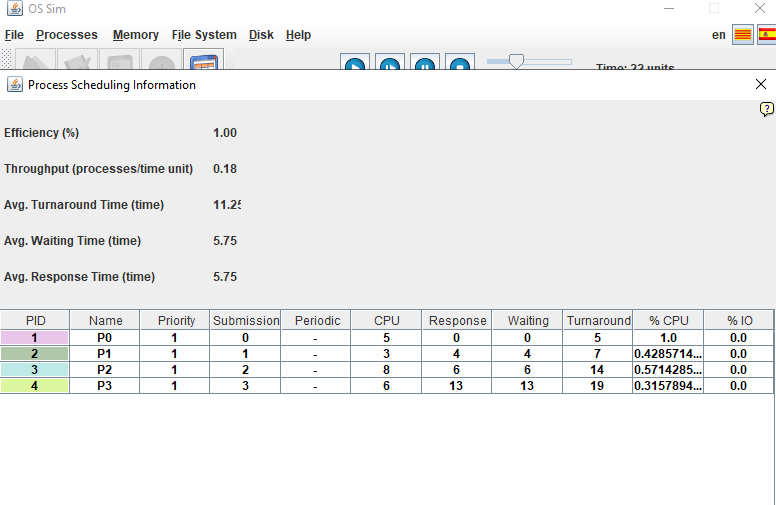


* + 1. Lakukan input proses sesuai dengan tabel berikut dengan memulai dengan P0 sebagai input proses yang pertama.





* + 1. Jika input sudah selesai dilakukan. Pilih tombol start pada bagian atas. Amati dan analisa proses yang terjadi.



* + 1. Isilah tabel berikut ini.

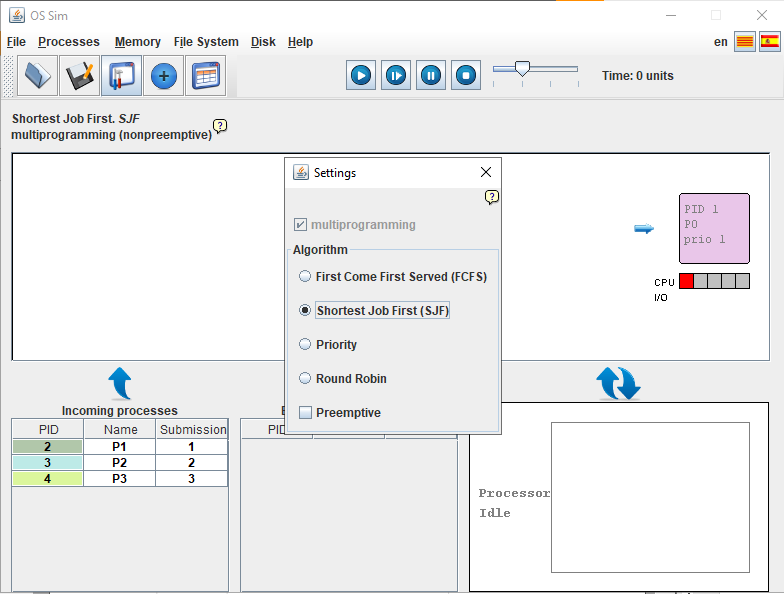
|  |  |
| --- | --- |
| **Process** | **Wait time : Service Time – Arrival Time** |
| **P0** | 0 |
| **P1** | 4 |
| **P2** | 6 |
| **P3** | 13 |
| **Av wait time** | 5.75 |

## Kesimpulan:

FCFS merupakan penjadwalan yang paling sederhana yang digunakan CPU. Dengan menggunakan algoritma ini setiap proses yang berada pada status ready dimasukkan kedalam FIFO queue atau antrian dengan prinsip first in first out, sesuai dengan waktu kedatangannya. Proses yang tiba terlebih dahulu yang akan dieksekusi.

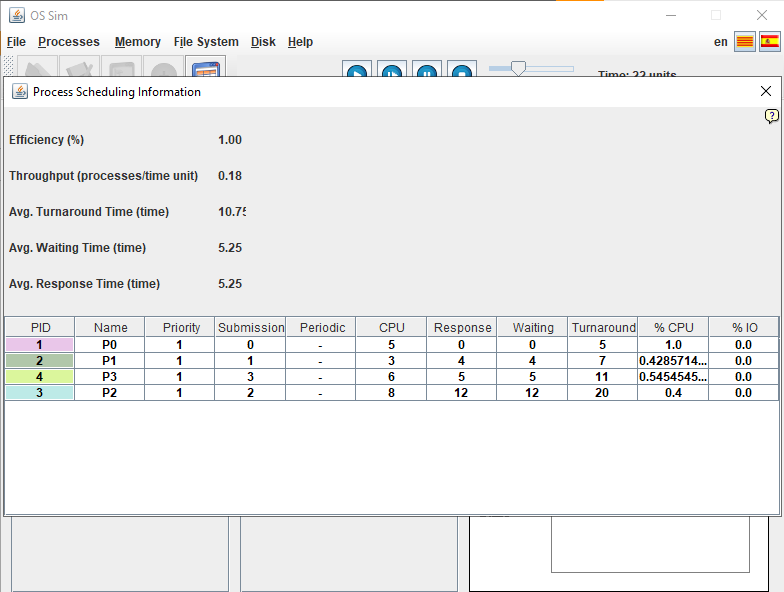
## Shortest Job First (SJF)

* + 1. Bukalah program OS Sim, selanjutnya pilih menu processes  process scheduling
    2. Selanjutnya pilihlah setting dan pilih algoritma Shortest Job First (SJF). Algoritma ini terdiri dari 2 jenis yaitu non-preemtive dan preemptive. Untuk mengaktifkan preemptive dengan mencentang menu tersebut. Sebaliknya jika menonaktifkan maka hanya cukup menghilangkan centangnya saja.

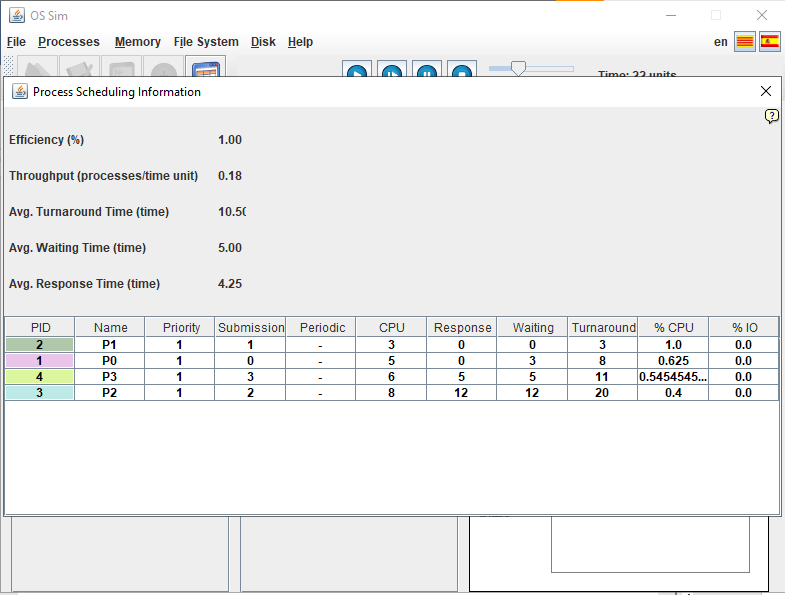


* + 1. Selanjutnya klik tombol start. Amati dan analisa proses yang terjadi. Lakukan perbandingan dari hasil keduanya.

## Non-Preemtive



**Preemtive**



* + 1. Isilah tabel berikut.

## Non-Preemtive

|  |  |
| --- | --- |
| **Process** | **Wait time : Service Time – Arrival Time** |
| **P0** | 0 |
| **P1** | 4 |
| **P2** | 5 |
| **P3** | 12 |
| **Av wait time** | 5.25 |

**Preemtive**

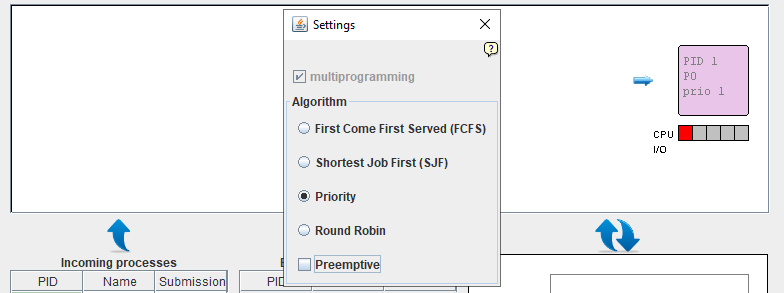
|  |  |
| --- | --- |
| **Process** | **Wait time : Service Time – Arrival Time** |
| **P0** | 0 |
| **P1** | 3 |
| **P2** | 5 |
| **P3** | 12 |
| **Av wait time** | 5.00 |

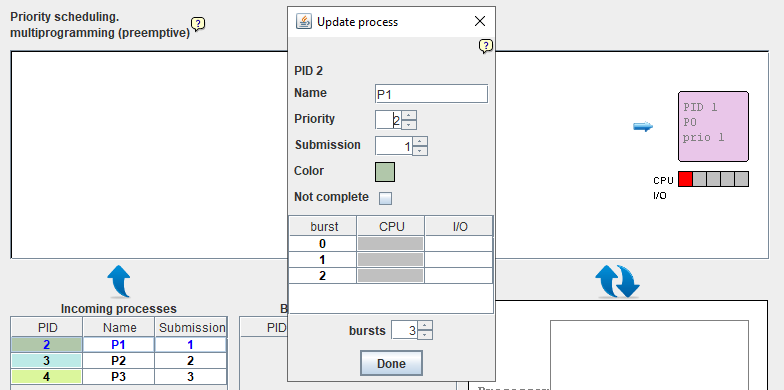
## Kesimpulan:

Pada algoritma ini setiap proses yang ada di ready queue akan dieksekusi berdasarkan burst time terkecil. Hal ini mengakibatkan waiting time yang pendek untuk setiap proses dan karena hal tersebut maka waiting time rata-ratanya juga menjadi pendek, sehingga dapat dikatakan bahwa algoritma ini adalah algoritma yang optimal.

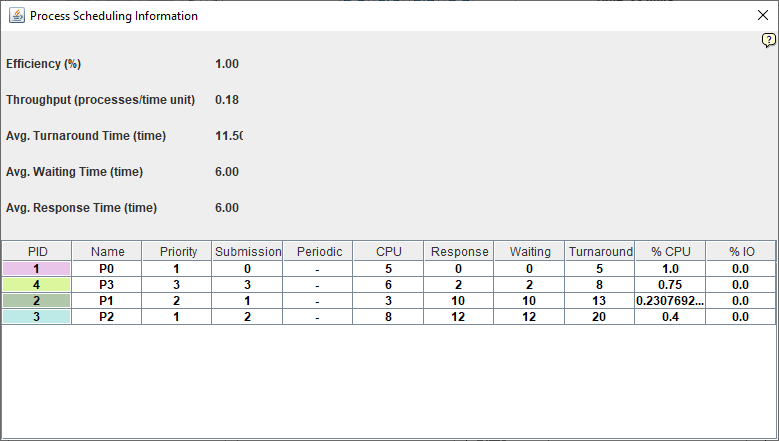
## Priority

* + 1. Pilihlah menu setting dan pilih algoritma Priority. Selanjutnya tambahkan priority pada setiap proses.





* + 1. Selanjutnya klik tombol start. Lakukan pengamatan dan analisa proses yang terjadi.Lengkapilah tabel berikut!



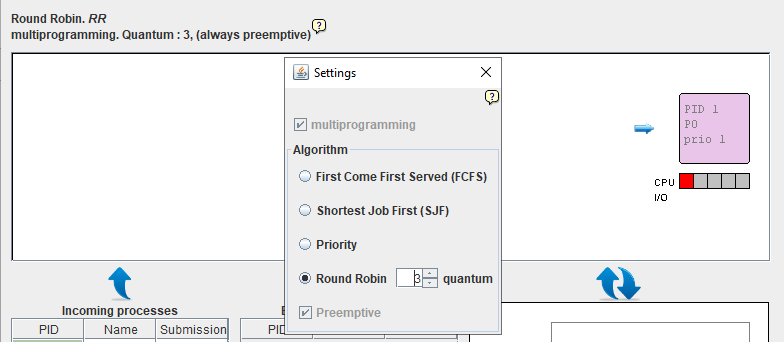
|  |  |
| --- | --- |
| **Process** | **Wait time : Service Time – Arrival Time** |
| **P0** | 0 |
| **P1** | 2 |
| **P2** | 10 |
| **P3** | 12 |
| **Av wait time** | 6.00 |

## Kesimpulan:

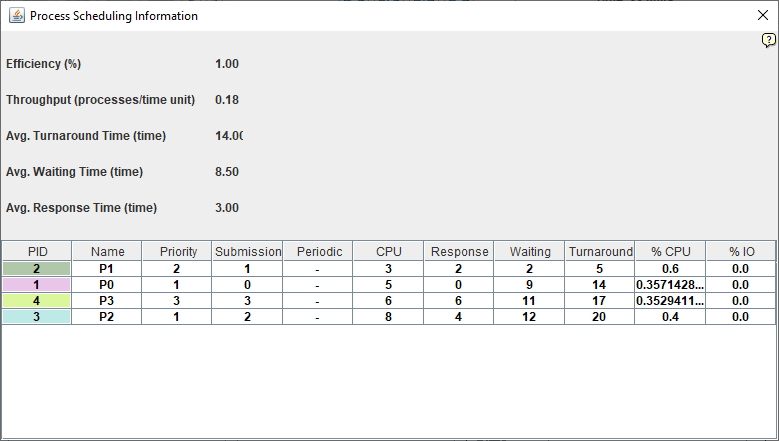
Priority Scheduling merupakan algoritma penjadwalan yang mendahulukan proses yang memiliki prioritas tertinggi. Setiap proses memiliki prioritasnya masing- masing.

## Round Robin

* + 1. Pilihlah menu setting dan pilih algoritma Round Robin. Selanjutnya tambahkan quantum time sebesar 3.



* + 1. Selanjutnya klik tombol start. Lakukan pengamatan dan analisa proses yang terjadi. Lengkapilah tabel berikut:



|  |  |
| --- | --- |
| **Process** | **Wait time : Service Time – Arrival Time** |
| **P0** | 2 |
| **P1** | 9 |
| **P2** | 11 |
| **P3** | 12 |
| **Av wait time** | 8.50 |

## Kesimpulan:

Algoritma ini menggilir proses yang ada di antrian. Proses akan mendapat jatah sebesar time quantum. Jika time quantum-nya habis atau proses sudah selesai, CPU akan dialokasikan ke proses berikutnya. Tentu proses ini cukup adil karena tak ada proses yang diprioritaskan, semua proses mendapat jatah waktu yang sama dari CPU yaitu (1/n), dan tak akan menunggu lebih lama dari (n-1)q dengan q adalah lama 1 quantum.

Algoritma ini sepenuhnya bergantung besarnya time quantum. Jika terlalu besar, algoritma ini akan sama saja dengan algoritma first come first served. Jika terlalu kecil, akan semakin banyak peralihan proses sehingga banyak waktu terbuang.